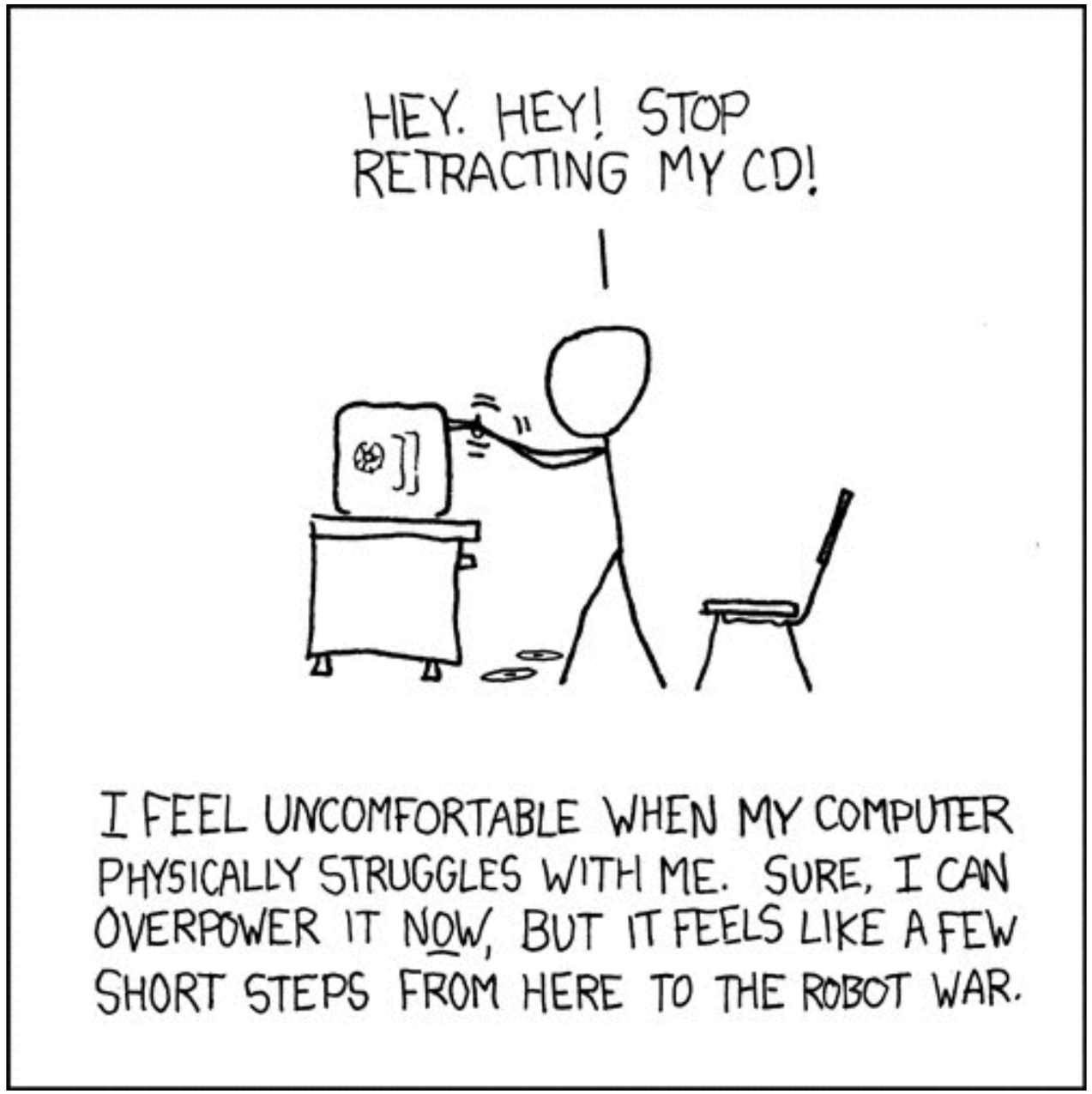
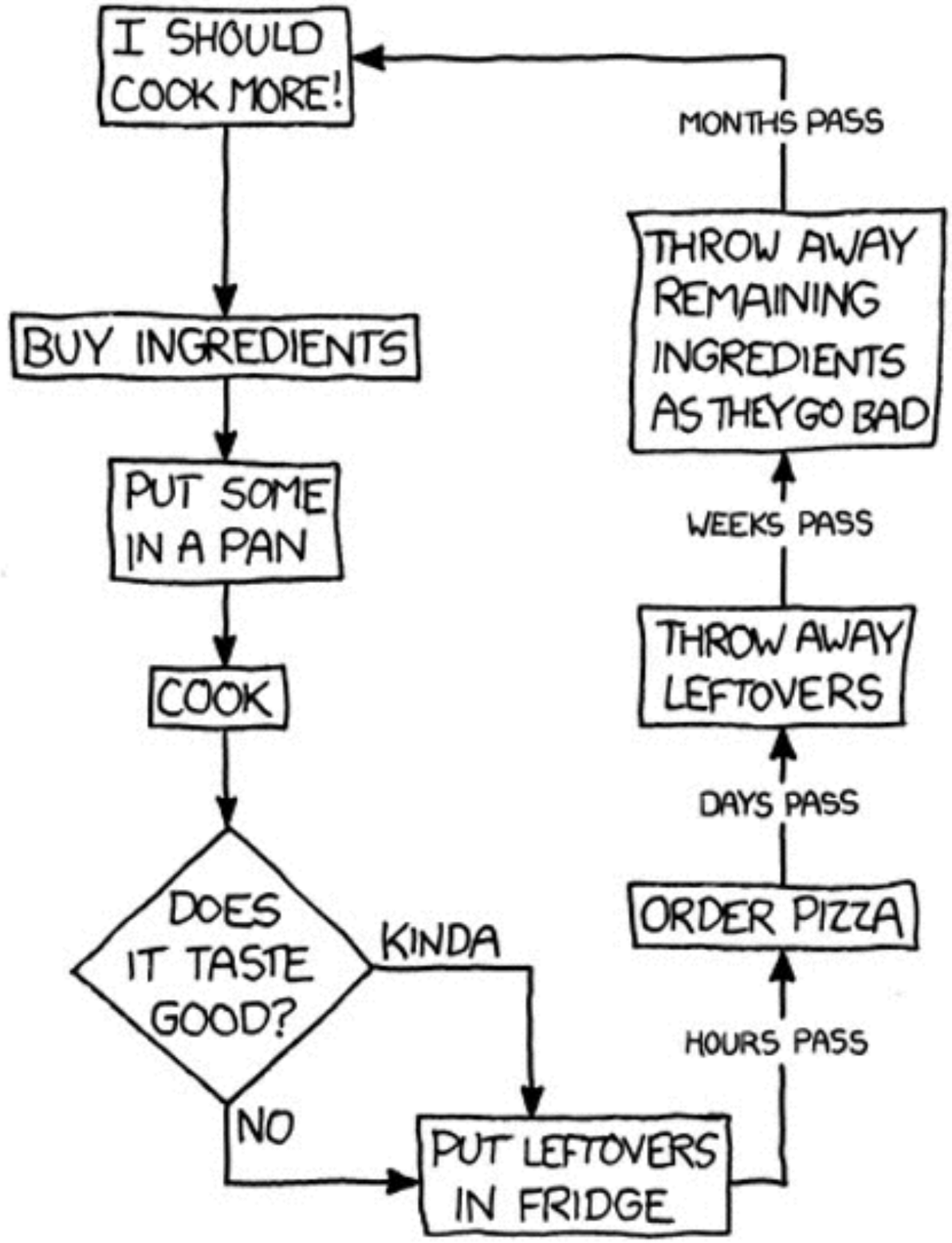

HEY. HEY! STOP RETRACTING MY CD!
I FEEL UNCOMFORTABLE WHEN MY COMPUTER PHYSICALLY STRUGGLES WITH ME. SURE, I CAN OVERPOWER IT NOW, BUT IT FEELS LIKE A FEW SHORT STEPS FROM HERE TO THE ROBOT WAR.

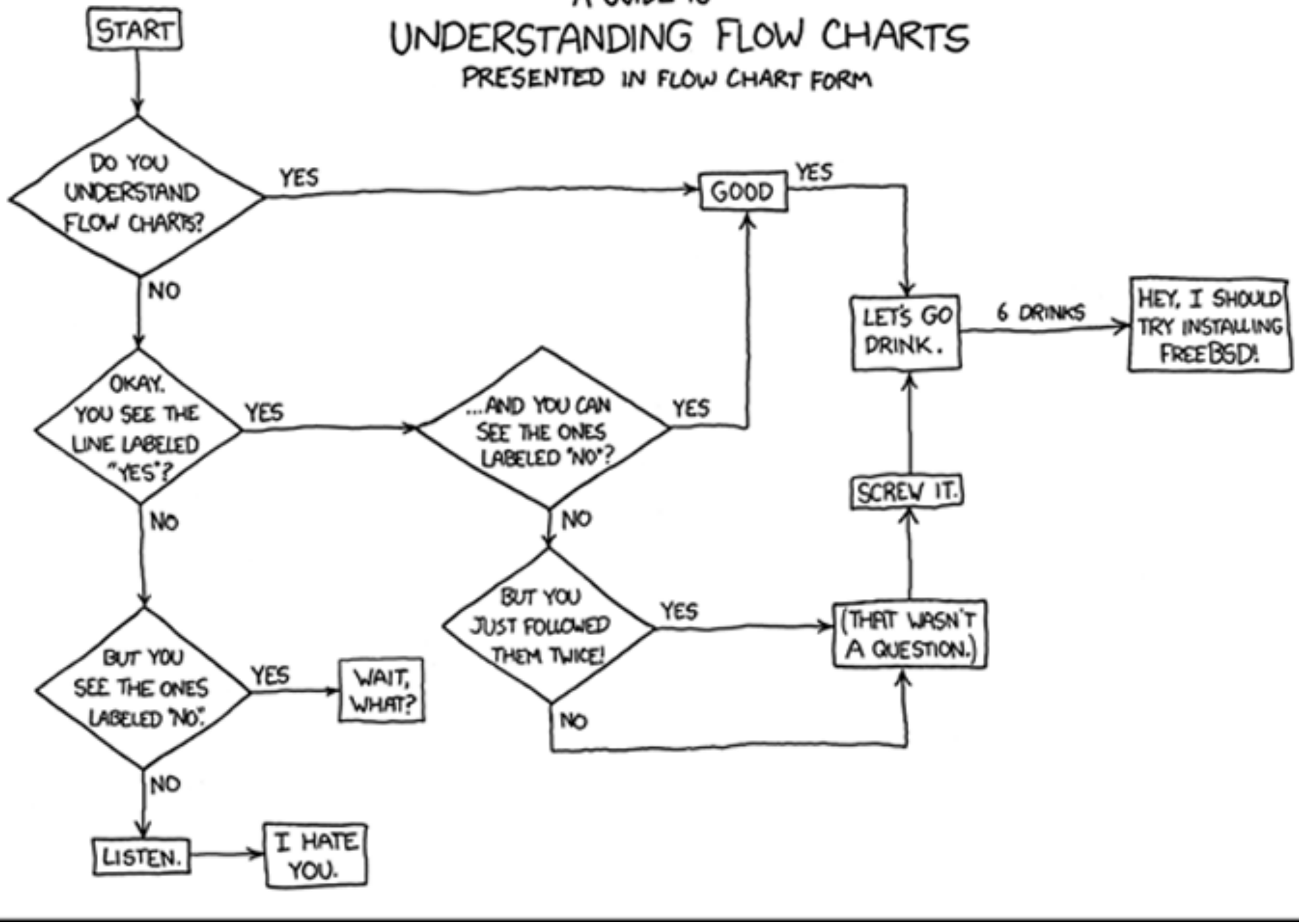
```
#!/bin/bash
```

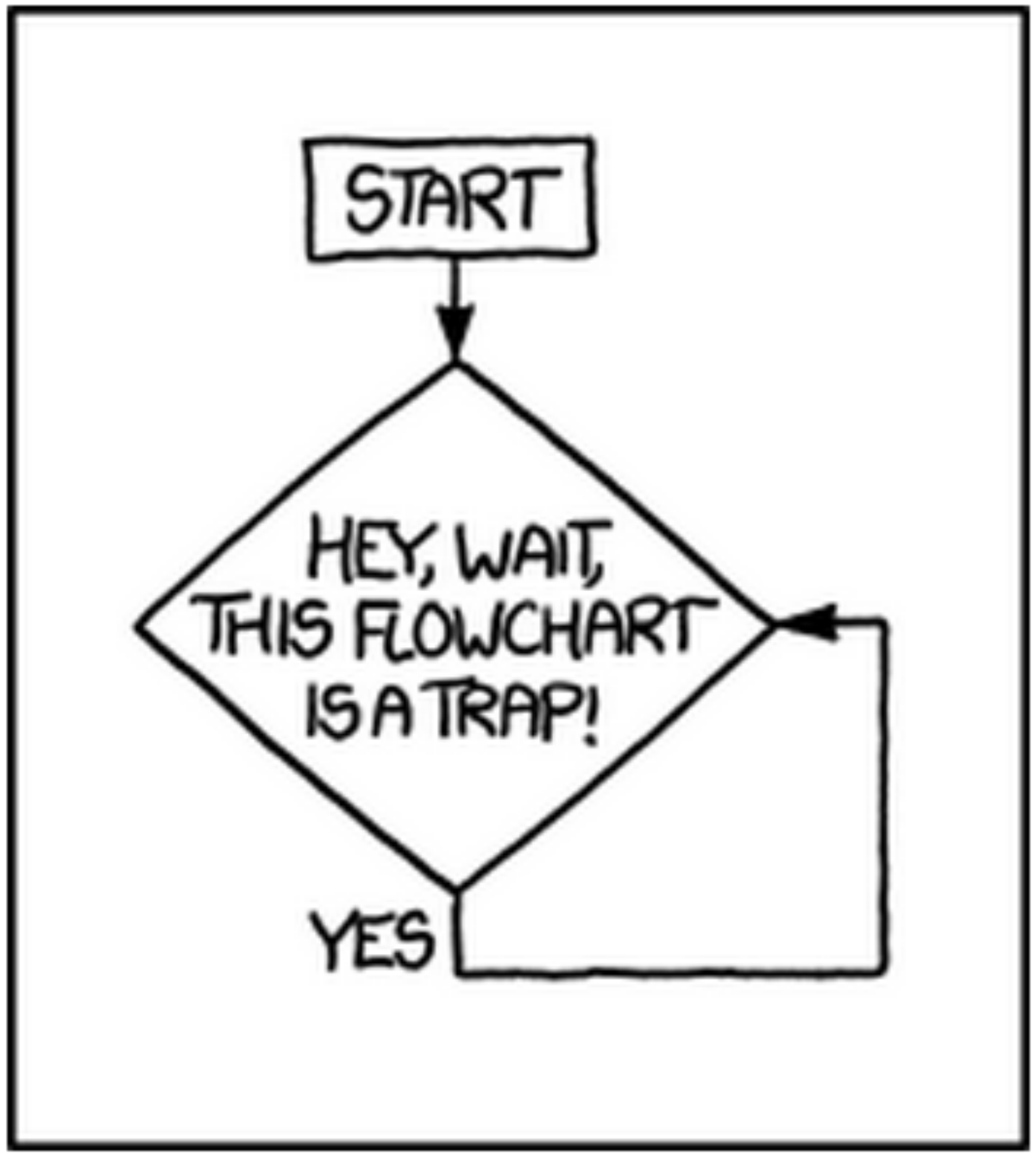
Learning to Cook



A GUIDE TO UNDERSTANDING FLOW CHARTS

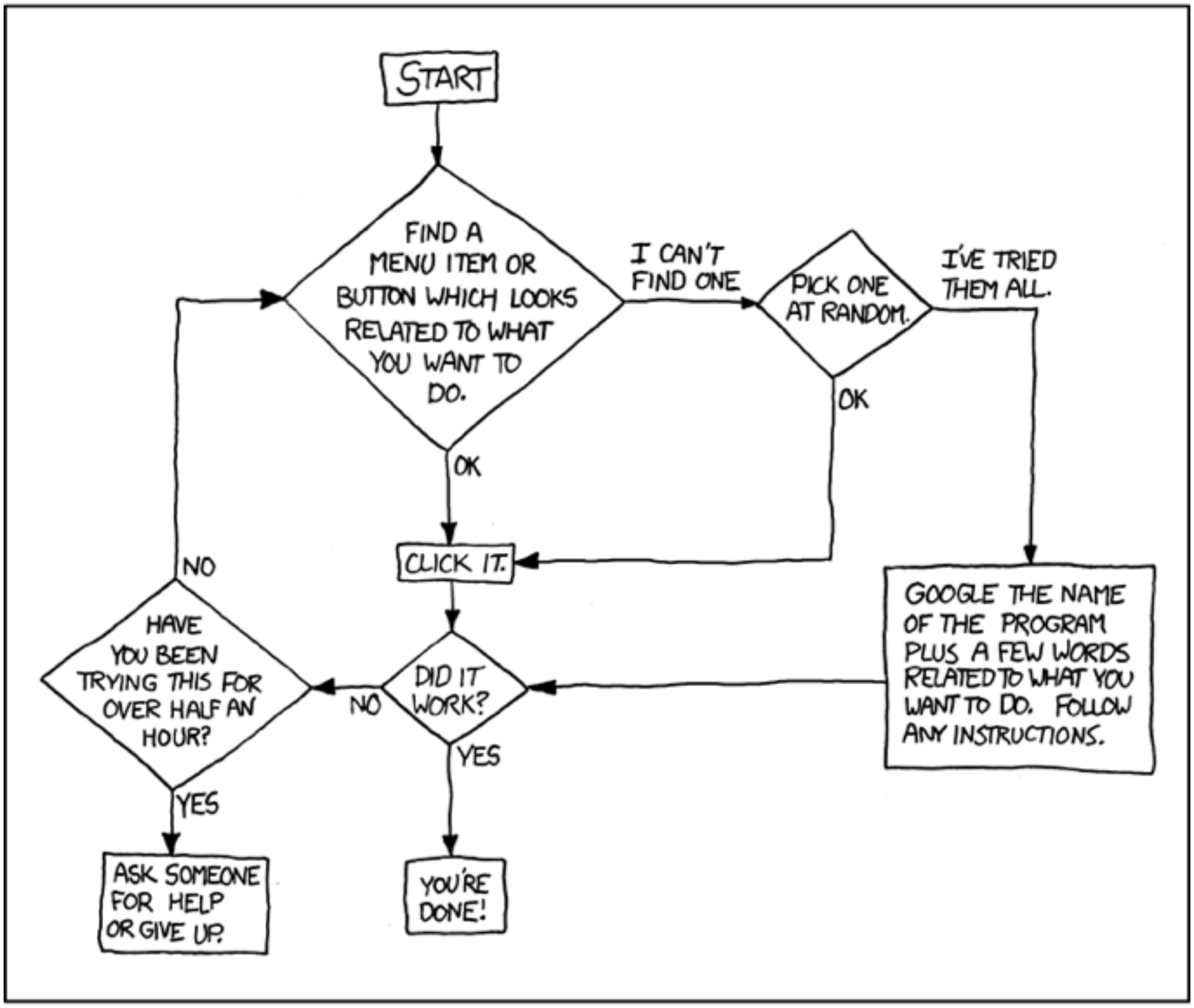
PRESENTED IN FLOW CHART FORM




START
HEY, WAIT, THIS FLOWCHART IS A TRAP!
YES

DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS, AND OTHER "NOT COMPUTER PEOPLE."

WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:



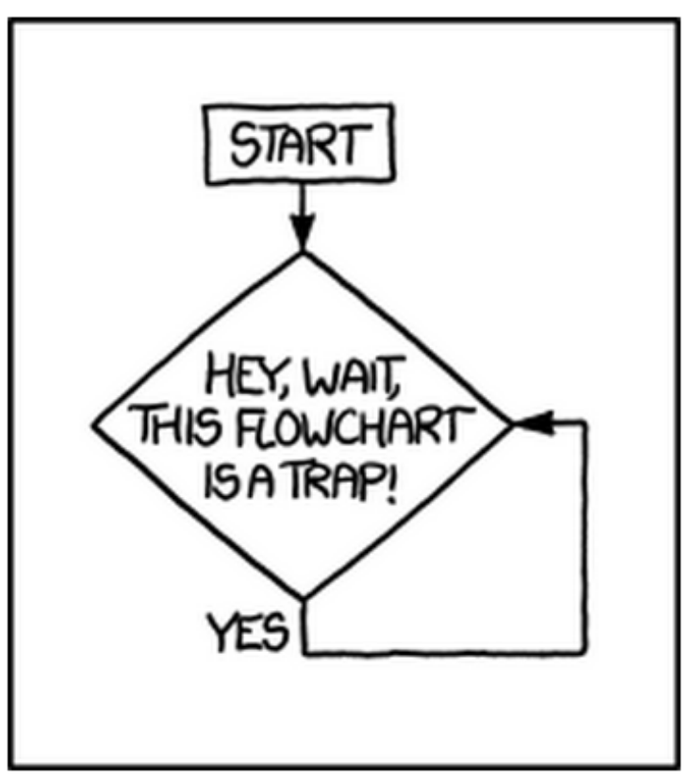
PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN. CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

```
> #!/bin/bash
```

Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa

De iteración:



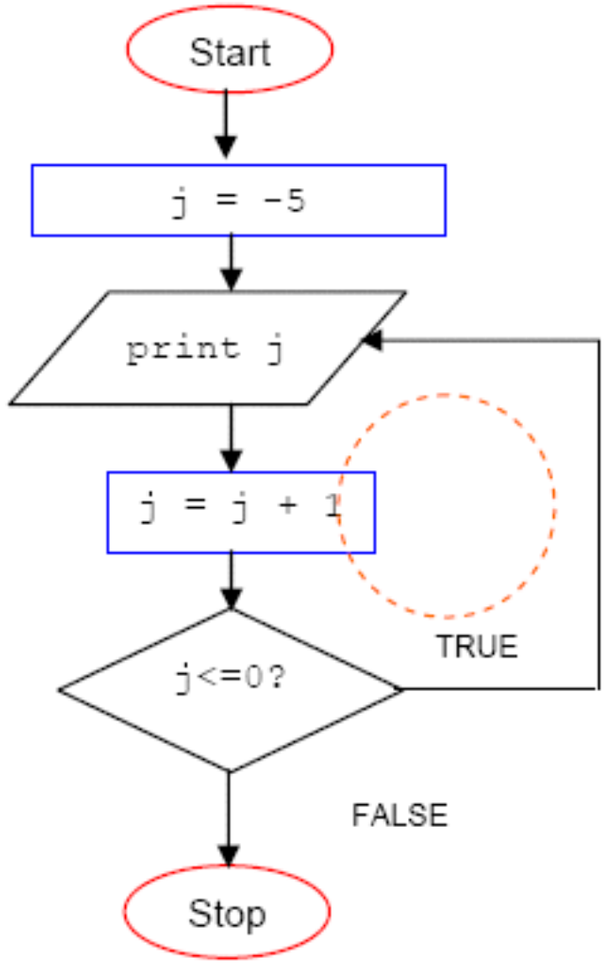

```
DO WHILE (Condición)
        (Bloque de sentencias)
LOOP
```

```
> #!/bin/bash
```

Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa

De iteración:




```
DO WHILE (Condición)
        (Bloque de sentencias)
LOOP
```

```
> #!/bin/bash
```

Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa

De iteración:

```
#!/bin/bash

x=1
while [ $x -le 5 ]
do
  echo "$x todavia es menor que 5"
  x=$(( $x + 1 ))
done
```

```
> #!/bin/bash
```

Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa

De iteración:

```
#!/bin/bash

x=1
until [ $x -ge 5 ]
do
  echo "$x todavia es menor que 5"
  x=$(( $x + 1 ))
done
```

¿Cuál es la diferencia?

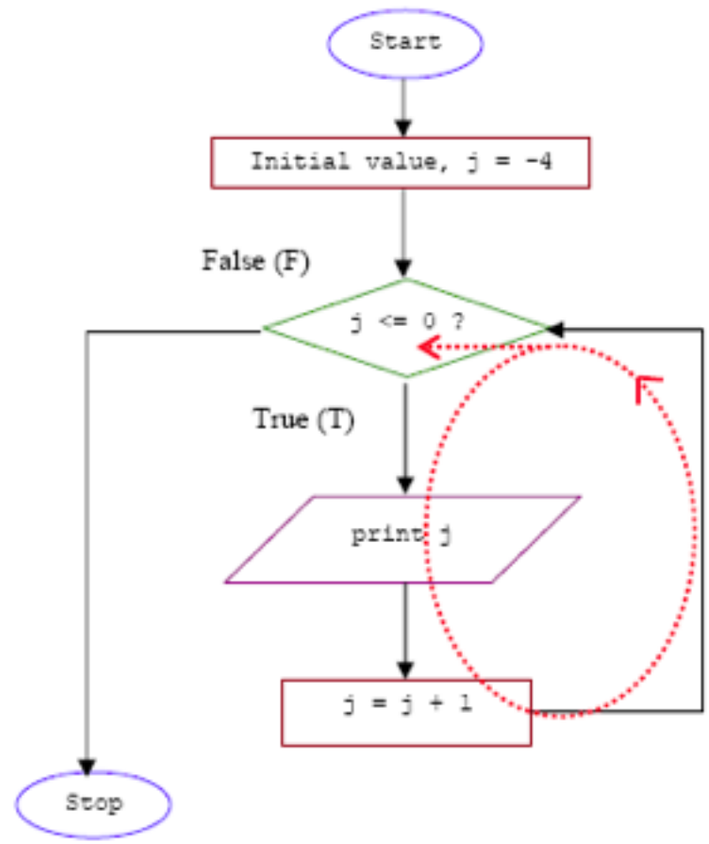
```
> #!/bin/bash
```

Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa

De iteración:

```
For (Variable) = (Expresión1) To (Expresión2) STEP (Salto)
    (Bloque de sentencias)
Next
```



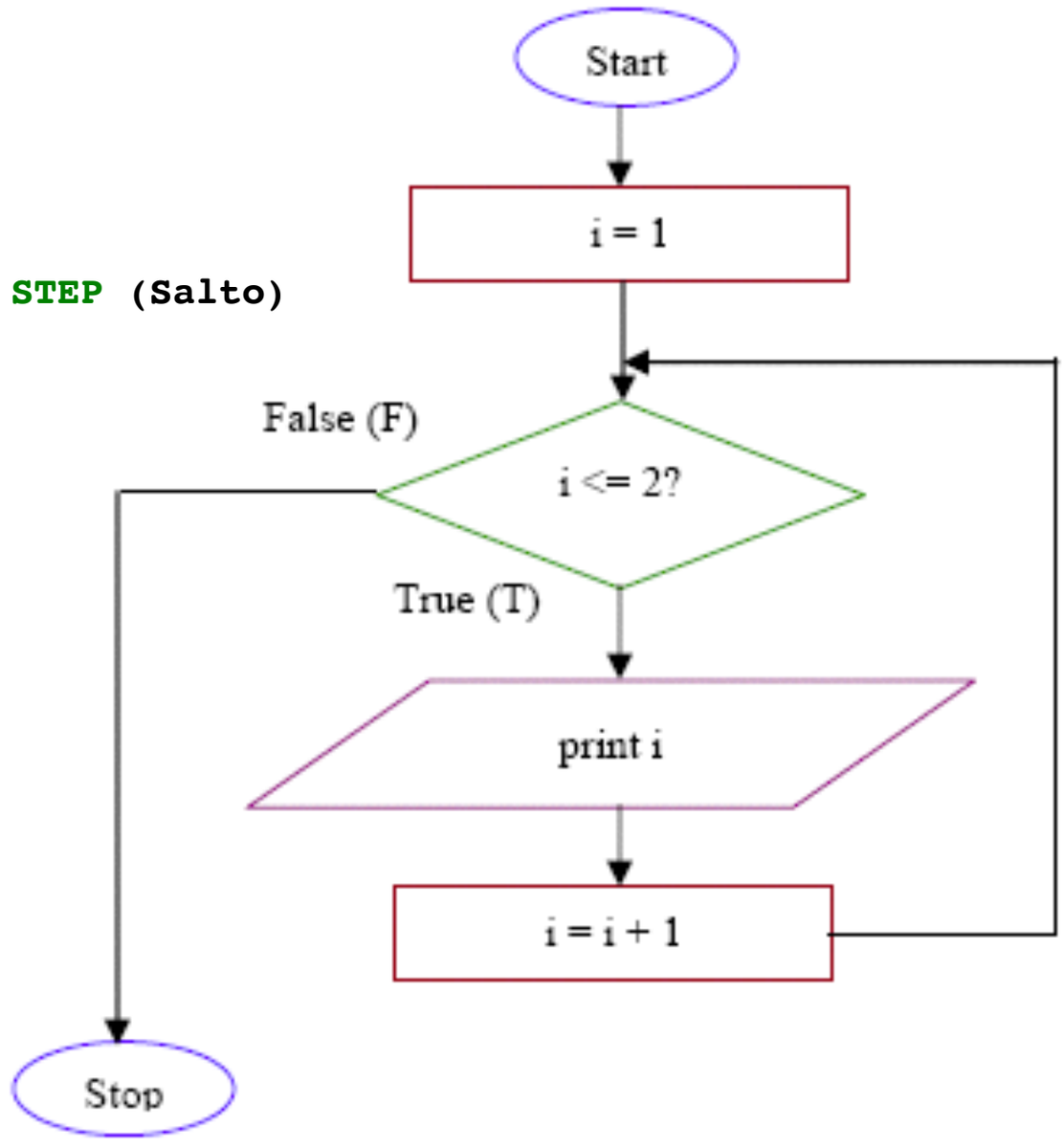
```
> #!/bin/bash
```

Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa

De iteración:

```
For (Variable) = (Expresión1) To (Expresión2) STEP (Salto)
      (Bloque de sentencias)
   Next
```



```
> #!/bin/bash
```

Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa

De iteración:

```
#!/bin/bash

contador=0

for i in $(ls)
do
   contador=$((contador+1))
   echo $i
done

echo "En total hay $contador archivos"
```

```
> #!/bin/bash
```

Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa

De selección:


```
IF (Condición) THEN
    (Bloque de sentencias 1)
ELSE
    (Bloque de sentencias 2)
END IF
```

```
> #!/bin/bash
```

Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa

De selección:

Como if evalua una condición, no es de mucho sentido que su argumento sea un comando. Los comandos siempre devuelven un estado (exit status) verdadero si el comando se realizó con éxito y falso de otro modo

```
$ls asd; echo $?

$ls ~; echo $?

$false; echo $?

$true; echo $?

$[ 4 = 1 ] ; echo $?

$[ 1 = 1 ] ; echo $?
```

Comando exitoso: 0
No exitoso: ≠ 0

> #!/bin/bash

Estructuras de Control

```
#!/bin/bash

FILE = $1
if [ -e "$FILE" ]; then
  if [ -f "$FILE" ]; then
    echo "$FILE es un archivo regular"
  fi
  if [ -d "$FILE" ]; then
    echo "$FILE es un directorio."
  fi
  if [ -r "$FILE" ]; then
    echo "$FILE es legible."
  fi
  if [ -w "$FILE" ]; then
    echo "$FILE es escribible."
  fi
  if [ -x "$FILE" ]; then
    echo "$FILE es ejecutable o navegable."
  fi
else
  echo "$FILE does not exist"
  exit 1
fi
exit
```

```
> #!/bin/bash
```

Estructuras de Control

Archivos:

Expresión	Verdadera si
file1 -nt file2	file1 es más nuevo que file2
file1 -ot file2	file1 es más nuevo que file2
-b file	file existe y es un dispositivo
-d file	file existe y es un directorio
-e file	file existe
-f file	file existe y es un archivo regular
-L file	file existe y es un link simbólico
-O file	file existe y el usuario es su dueño
-r file	file existe y es legible
-r file	file existe y tiene un tamaño maor a cero
-w file	file existe y es escribible
-x file	file existe y es ejecutable

Cadenas y enteros:

Expresión	Verdadera si
string	No es null
-n string	La longitud es diferente de cero
-z string	La longitud es igual a cero
string1 == string 2	Son iguales
string1 != string2	Son diferentes
string1 > string2	string1 organiza primero que string2
string2 < string2	string2 organiza primero que string1
int1 -eq int2	Iguales
int1 -ne int2	Diferentes
int1 -le int2	int1 menor o igual a int2
int1 -lt int2	int1 menor que int2
int1 -ge int2	int1 mayor o igual a int2
int1 -gt int2	int1 mayor que int2

> #!/bin/bash

QUIZ!

Haga un diagrama de flujo para el siguiente script de bash. ¿Qué hace este script?

```
#!/bin/bash

suma=0

for i in $(seq 1 1 100)
do
    if [ $i -le 50]; then
        suma=$(($suma+$i))
    fi
done

echo “El resultado es $suma”
```